AFFIDAVIT

People's Republic of China)
Guangdong Province)
Guangzhou) ss:
Consulate General of the)
United States of America)

Before me, David H. Kornhauser, Vice Consul of the United States of America at Guangzhou, duly commissioned and qualified, personally appeared Yang Gao , who, being duly sworn, deposes and says:

Please see the attached.

Subscribed and sworn to before me
this 30th day of July 2004

Affidavit

In re first Office Action for Application Number 09/916,252:

This reply concerns the rejection of patent application 09/916,252 based on Erik Hatcher's anticipation, "Remote Scripting Using a Servlet".

Hatcher's prior art, it is claimed in the Office Action, receives the output of an HTTP response associated with an HTTP request into an HTML iframe element **without adding a Universal Resource Locator (URL) to the application's history list.**

In fact, Appeon can demonstrate that the prior art **does** add the URL to the application's history list.

Attached are two sets of source code:

- Erik Hatcher's source code unchanged from the first office action on our application.
- Appeon source code. To demonstrate the Appeon solution, the same user interface code was used, with our modifications to the remote procedure call based on our different approach as disclosed in our patent application.

Appeon's different solution does not add the URL to the application's history list and therefore is a different solution, not anticipated by prior art.

Sincerely

Yang Gao
Chief Technology Officer
Appeon Corporation

July 30th. 2004.

Our Source Code.

File: appeonRPC2.htm 2004-8-5, 10:28:12


```
<html>
<script>
var ifrCount=0
function _I(callback, param)
{
                var strParam = unescape(param);
                window.status="";
                eval(callback + "('" + strParam + "')");
}

function LMLPost(serverPage, callback, serverFunction, data)
{
                var strHTML=new String();
                if(!document.forms.LMLform)
        {
                strHTML="<div><form name=\"LMLform\"method=\"post\"action=\""+ serverPage + "\">";
                strHTML+="<input type=\"hidden\" name=\"func\">";
                strHTML+="<input type=\"hidden\" name=\"params\">";
                strHTML+="<input type=\"hidden\" name=\"callback\">";
                strHTML+="</form></div>";
                document.body.insertAdjacentHTML("BeforeEnd",strHTML)
        }

        if(ifrCount>0)
                document.all('div_'+(ifrCount-1)).innerHTML='';
                document.forms.LMLform.func.value = serverFunction;
                document.forms.LMLform.params.value = data;
                document.forms.LMLform.callback.value = callback;
      strHTML="<div id='div_"+ifrCount+"'><iframe style=\"position:absolute;visibility:hidden;left:0;top:0\" name
= '" + ifrCount +"'></iframe></div>";
                document.body.insertAdjacentHTML("BeforeEnd",strHTML);
                document.forms.LMLform.target="+ifrCount;
                document.forms.LMLform.submit();
                ifrCount++;
}

function post_data()
{

      var oTemp = document.all("category");
      var nIndex = oTemp.options[oTemp.selectedIndex].value;
      LMLPost("server2.asp", "my_callback", "getData",nIndex);
}

function clearDropDown (selField)
 {
   while (selField.options.length > 0)
     selField.options[0] = null;
 }

 function my_callback (valueTextStr)
 {

  if(valueTextStr == "error")
  {
          alert("error testing: call server failed");
          return;
  }
   var selField = document.all("subcategory");
   clearDropDown(selField);

```

```
62    // create an array of each item for the dropdown
63    var aPairs = valueTextStr.split(";");
64    if (valueTextStr.substr(valueTextStr.length - 1) == ';') {
65       aPairs[aPairs.length - 1] = null;
66       aPairs.length--;
67    }
68    // Fill 'er up
69    for (var i=0; i < aPairs.length; i++) {
70     aValueText = aPairs[i].split(",");  // each item is a "value,name" pair
71     oItem = new Option;
72     oItem.value = aValueText[0];
73     oItem.text = aValueText[1];
74     selField.options[selField.options.length] = oItem;
75    }
76    selField.options.selectedIndex = 0;
77  }
78 </script>
79 <body onLoad="javascript:post_data()">
80 <TABLE>
81    <TR>
82     <TH>Remote Scripting Type:</TH>
83     <TD>
84       <input type="radio" name="clientType" value="JSRS" CHECKED>AppeonRPC
85     </TD>
86    </TR>
87    <TR>
88     <TH>Category:</TH>
89     <TD>
90       <SELECT name="category" onChange="javascript:post_data()">
91         <OPTION value="0" SELECTED>Category 0</OPTION>
92         <OPTION value="1">Category 1</OPTION>
93         <OPTION value="2">Category 2</OPTION>
94         <OPTION value="3">Error Test</OPTION>
95       </SELECT>
96     </TD>
97    </TR>
98    <TR>
99     <TH>Subcategory:</TH>
100    <TD>
101      <SELECT name="subcategory">
102        <!-- Need a placeholder until it can get loaded -->
103        <OPTION value="-1" SELECTED>----------</OPTION>
104      </SELECT>
105    </TD>
106   </TR>
107  </TABLE>
108 </body>
109 </html>
```

```
1
2  <!--#include file="serverproxy2.inc"-->
3  <script runat=Server language=VBScript>
4  function getData(strPara)
5
6      getData = strPara
7  end function
8  </script>
9
```

```
<%@ Language=JavaScript %>
<%
   var _LML_PARAMS_DELIM = String.fromCharCode(1);
   var i, strBody, strResult;
   var strFunc, strCallback;
   var strParams, arrParams;
    strFunc = Request("func").item;
   strParams = Request("params").item;
   strCallback = Request("callback").item;
   response.write(strFunc+"<br>"+strCallback+"<br>"+strParams);

   strParams = getSubcategories(strParams);
   strParams = "\"" + strParams + "\"";
   strResult = escape(eval(strFunc + "(" + strParams + ")"));
   strBody = "<HTML><HEAD></HEAD>"
   strBody += "<BODY onload=parent.window._l(";
   strBody += "\"" + strCallback + "\",";
   strBody += "\"" + strResult + "\"";
   strBody += ")>";
   strBody += "</BODY></HTML>";
   Response.Write(strBody);
   function EscapeSpecialChars(strSrc)
   {

      var strDest = strSrc;
      strDest = strDest.replace(/\\/g, '\\\\');
      strDest = strDest.replace(/\r/g, '\\r');
      strDest = strDest.replace(/\n/g, '\\n');
      strDest = strDest.replace(/"/g, '\\"');
      strDest = strDest.replace(/'/g, '\\"');
      return strDest;

      return strSrc;
   }
 function getSubcategories(strIndex)
   {
            var  retval;

            switch(strIndex)
            {
            case "0":
                  retval = "0,Category 0 - Subcategory 0;1,Category 0 - Subcategory 1;2,Category 0 -
Subcategory 2;";
                  break;
            case "1":
                  retval = "0,Category 1 - Subcategory 0;1,Category 1 - Subcategory 1;2,Category 1 -
Subcategory 2;";
                  break;
            case "2":
                  retval = "0,Category 2 - Subcategory 0;1,Category 2 - Subcategory 1;2,Category 2 -
Subcategory 2;";
                  break;
            case "3":
                  retval = "error";
                  break;
            }
            return retval;

   }
%>
```

Erik Hatcher's source code.


```
<HTML>

<HEAD>
<title>Remote Scripting w/ Servlet Example</title>
<script language="javascript" src="/jsrsClient.js"></script>
<script language=Javascript src="_ScriptLibrary/rs.htm"></script>

<script language=javascript>
 function clearDropDown (selField)
 {
  while (selField.options.length > 0)
   selField.options[0] = null;
 }

 function populateDropDown (valueTextStr)
 {
  //debugger;
  var selField = document.form1.subcategory;
  clearDropDown(selField);

  // create an array of each item for the dropdown
  var aPairs = valueTextStr.split(";");
  if (valueTextStr.substr(valueTextStr.length - 1) == ';') {
     aPairs[aPairs.length - 1] = null;
     aPairs.length--;
  }
  // Fill 'er up
  for (var i=0; i < aPairs.length; i++) {
   aValueText = aPairs[i].split(",");  // each item is a "value,name" pair
   oItem = new Option;
   oItem.value = aValueText[0];
   oItem.text = aValueText[1];
   selField.options[selField.options.length] = oItem;
  }
  selField.options.selectedIndex = 0;
 }

 function categoryChanged()
 {
    // JSRS
    jsrsExecute("/myservlet/RSExample", populateDropDown, "getSubcategories",
document.form1.category.options[document.form1.category.selectedIndex].value);
 }
</script>

</HEAD>
<BODY onLoad="javascript:categoryChanged()">
<FORM name="form1">
 <TABLE>
  <TR>
   <TH>Remote Scripting Type:</TH>
   <TD>
    <input type="radio" name="clientType" value="JSRS" CHECKED>JSRS
   </TD>
  </TR>
  <TR>
   <TH>Category:</TH>
   <TD>
    <SELECT name="category" onChange="javascript:categoryChanged()">
     <OPTION value="0" SELECTED>Category 0</OPTION>
     <OPTION value="1">Category 1</OPTION>
     <OPTION value="2">Category 2</OPTION>
```

```
62          <OPTION value="3">Error Test</OPTION>
63        </SELECT>
64      </TD>
65    </TR>
66    <TR>
67      <TH>Subcategory:</TH>
68      <TD>
69        <SELECT name="subcategory">
70          <!-- Need a placeholder until it can get loaded -->
71          <OPTION value="-1" SELECTED>----------------------------------------------------------------------------------------</OPTION>
72        </SELECT>
73      </TD>
74    </TR>
75  </TABLE>
76 </FORM>
77 </BODY>
78 </HTML>
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
```

```
1  //
2  // jsrsClient.js - javascript remote scripting client include
3  //
4  // Author:  Brent Ashley [jsrs@megahuge.com]
5  //
6  // make asynchronous remote calls to server without client page refresh
7  //
8  // see license.txt for copyright and license information
9
10 /*
11 see history.txt for full history
12 2.0  26 Jul 2001 - added POST capability for IE/MOZ
13 2.2  10 Aug 2003 - added Opera support
14 2.3(beta)  10 Oct 2003 - added Konqueror support - **needs more testing**
15 */
16
17 // callback pool needs global scope
18 var jsrsContextPoolSize = 0;
19 var jsrsContextMaxPool = 10;
20 var jsrsContextPool = new Array();
21 var jsrsBrowser = jsrsBrowserSniff();
22 var jsrsPOST = true;
23 var containerName;
24
25 // constructor for context object
26 function jsrsContextObj( contextID ){
27
28   // properties
29   this.id = contextID;
30   this.busy = true;
31   this.callback = null;
32   this.container = contextCreateContainer( contextID );
33
34   // methods
35   this.GET = contextGET;
36   this.POST = contextPOST;
37   this.getPayload = contextGetPayload;
38   this.setVisibility = contextSetVisibility;
39 }
40
41 //  method functions are not privately scoped
42 //  because Netscape's debugger chokes on private functions
43 function contextCreateContainer( containerName ){
44   // creates hidden container to receive server data
45   var container;
46   switch( jsrsBrowser ) {
47     case 'NS':
48       container = new Layer(100);
49       container.name = containerName;
50       container.visibility = 'hidden';
51       container.clip.width = 100;
52       container.clip.height = 100;
53       break;
54
55     case 'IE':
56       document.body.insertAdjacentHTML( "afterBegin", '<span id="SPAN' + containerName + '"></span>' );
57       var span = document.all( "SPAN" + containerName );
58       var html = '<iframe name="' + containerName + '" src=""></iframe>';
59       span.innerHTML = html;
60       span.style.display = 'none';
61       container = window.frames[ containerName ];
62       break;
```

```
63
64      case 'MOZ':
65        var span = document.createElement('SPAN');
66        span.id = "SPAN" + containerName;
67        document.body.appendChild( span );
68        var iframe = document.createElement('IFRAME');
69        iframe.name = containerName;
70        iframe.id = containerName;
71        span.appendChild( iframe );
72        container = iframe;
73        break;
74
75      case 'OPR':
76        var span = document.createElement('SPAN');
77        span.id = "SPAN" + containerName;
78        document.body.appendChild( span );
79        var iframe = document.createElement('IFRAME');
80        iframe.name = containerName;
81        iframe.id = containerName;
82        span.appendChild( iframe );
83        container = iframe;
84        break;
85
86      case 'KONQ':
87        var span = document.createElement('SPAN');
88        span.id = "SPAN" + containerName;
89        document.body.appendChild( span );
90        var iframe = document.createElement('IFRAME');
91        iframe.name = containerName;
92        iframe.id = containerName;
93        span.appendChild( iframe );
94        container = iframe;
95
96        // Needs to be hidden for Konqueror, otherwise it'll appear on the page
97        span.style.display = none;
98        iframe.style.display = none;
99        iframe.style.visibility = hidden;
100       iframe.height = 0;
101       iframe.width = 0;
102
103       break;
104   }
105   return container;
106 }
107
108 function contextPOST( rsPage, func, parms ){
109
110 //debugger;
111   var d = new Date();
112   var unique = d.getTime() + " + Math.floor(1000 * Math.random());
113   var doc = (jsrsBrowser == "IE" ) ? this.container.document : this.container.contentDocument;
114   doc.open();
115   doc.write('<html><body>');
116   doc.write('<form name="jsrsForm" method="post" target="" ');
117   doc.write(' action="' + rsPage + '?U=' + unique + '">');
118   doc.write('<input type="hidden" name="C" value="' + this.id + '">');
119
120   // func and parms are optional
121   if (func != null){
122   doc.write('<input type="hidden" name="F" value="' + func + '">');
123
124     if (parms != null){
```

```
125       if (typeof(parms) == "string"){
126         // single parameter
127         doc.write( '<input type="hidden" name="P0" '
128               + 'value="[' + jsrsEscapeQQ(parms) + ']">');
129       } else {
130         // assume parms is array of strings
131         for( var i=0; i < parms.length; i++ ){
132           doc.write( '<input type="hidden" name="P' + i + '" '
133                 + 'value="[' + jsrsEscapeQQ(parms[i]) + ']">');
134         }
135       } // parm type
136     } // parms
137   } // func
138
139   doc.write('</form></body></html>');
140   doc.close();
141   doc.forms['jsrsForm'].submit();
142 }
143
144 function contextGET( rsPage, func, parms ){
145 //debugger;
146   // build URL to call
147   var URL = rsPage;
148
149   // always send context
150   URL += "?C=" + this.id;
151
152   // func and parms are optional
153   if (func != null){
154     URL += "&F=" + escape(func);
155
156     if (parms != null){
157       if (typeof(parms) == "string"){
158         // single parameter
159         URL += "&P0=[" + escape(parms+") + "]";
160       } else {
161         // assume parms is array of strings
162         for( var i=0; i < parms.length; i++ ){
163           URL += "&P" + i + "=[" + escape(parms[i]+") + "]";
164         }
165       } // parm type
166     } // parms
167   } // func
168
169   // unique string to defeat cache
170   var d = new Date();
171   URL += "&U=" + d.getTime();
172
173   // make the call
174   switch( jsrsBrowser ) {
175     case 'NS':
176       this.container.src = URL;
177       break;
178     case 'IE':
179       this.container.document.location.replace(URL);
180       break;
181     case 'MOZ':
182       this.container.src = '';
183       this.container.src = URL;
184       break;
185     case 'OPR':
186       this.container.src = '';
```

```
187       this.container.src = URL;
188       break;
189     case 'KONQ':
190       this.container.src = '';
191       this.container.src = URL;
192       break;
193   }
194 }
195
196 function contextGetPayload(){
197   switch( jsrsBrowser ) {
198     case 'NS':
199       return this.container.document.forms['jsrs_Form'].elements['jsrs_Payload'].value;
200     case 'IE':
201       return this.container.document.forms['jsrs_Form']['jsrs_Payload'].value;
202     case 'MOZ':
203       return window.frames[this.container.name].document.forms['jsrs_Form']['jsrs_Payload'].value;
204     case 'OPR':
205       var textElement = window.frames[this.container.name].document.getElementById("jsrs_Payload");
206     case 'KONQ':
207       var textElement = window.frames[this.container.name].document.getElementById("jsrs_Payload");
208       return textElement.value;
209   }
210 }
211
212 function contextSetVisibility( vis ){
213   switch( jsrsBrowser ) {
214     case 'NS':
215       this.container.visibility = (vis)? 'show' : 'hidden';
216       break;
217     case 'IE':
218       document.all("SPAN" + this.id ).style.display = (vis)? '' : 'none';
219       break;
220     case 'MOZ':
221       document.getElementById("SPAN" + this.id).style.visibility = (vis)? '' : 'hidden';
222     case 'OPR':
223       document.getElementById("SPAN" + this.id).style.visibility = (vis)? '' : 'hidden';
224       this.container.width = (vis)? 250 : 0;
225       this.container.height = (vis)? 100 : 0;
226       break;
227   }
228 }
229
230 // end of context constructor
231
232 function jsrsGetContextID(){
233   var contextObj;
234   for (var i = 1; i <= jsrsContextPoolSize; i++){
235     contextObj = jsrsContextPool[ 'jsrs' + i ];
236     if ( !contextObj.busy ){
237       contextObj.busy = true;
238       return contextObj.id;
239     }
240   }
241   // if we got here, there are no existing free contexts
242   if ( jsrsContextPoolSize <= jsrsContextMaxPool ){
243     // create new context
244     var contextID = "jsrs" + (jsrsContextPoolSize + 1);
245     jsrsContextPool[ contextID ] = new jsrsContextObj( contextID );
246     jsrsContextPoolSize++;
247     return contextID;
248   } else {
```

```
249     alert( "jsrs Error:  context pool full" );
250     return null;
251   }
252 }
253
254 function jsrsExecute( rspage, callback, func, parms, visibility ){
255   // call a server routine from client code
256   //
257   // rspage      - href to asp file
258   // callback    - function to call on return
259   //               or null if no return needed
260   //               (passes returned string to callback)
261   // func        - sub or function name  to call
262   // parm         - string parameter to function
263   //               or array of string parameters if more than one
264   // visibility  - optional boolean to make container visible for debugging
265
266   // get context
267  //debugger;
268   var contextObj = jsrsContextPool[ jsrsGetContextID() ];
269   contextObj.callback = callback;
270
271   var vis = (visibility == null)? false : visibility;
272   contextObj.setVisibility( vis );
273
274   if ( jsrsPOST && ((jsrsBrowser == 'IE') || (jsrsBrowser == 'MOZ'))){
275     contextObj.POST( rspage, func, parms );
276   } else {
277     contextObj.GET( rspage, func, parms );
278   }
279
280   return contextObj.id;
281 }
282
283 function jsrsLoaded( contextID ){
284   // get context object and invoke callback
285   var contextObj = jsrsContextPool[ contextID ];
286   if( contextObj.callback != null){
287     contextObj.callback( jsrsUnescape( contextObj.getPayload() ), contextID );
288   }
289   // clean up and return context to pool
290   contextObj.callback = null;
291   contextObj.busy = false;
292 }
293
294 function jsrsError( contextID, str ){
295   alert( unescape(str) );
296   jsrsContextPool[ contextID ].busy = false
297 }
298
299 function jsrsEscapeQQ( thing ){
300   return thing.replace(/"'/g, '\\"');
301 }
302
303 function jsrsUnescape( str ){
304   // payload has slashes escaped with whacks
305   return str.replace( /\\/g, "/" );
306 }
307
308 function jsrsBrowserSniff(){
309   if (document.layers) return "NS";
310   if (document.all) {
```

```
311             // But is it really IE?
312             // convert all characters to lowercase to simplify testing
313             var agt=navigator.userAgent.toLowerCase();
314             var is_opera = (agt.indexOf("opera") != -1);
315             var is_konq = (agt.indexOf("konqueror") != -1);
316             if(is_opera) {
317                 return "OPR";
318             } else {
319                 if(is_konq) {
320                     return "KONQ";
321                 } else {
322                     // Really is IE
323                     return "IE";
324                 }
325             }
326   }
327   if (document.getElementById) return "MOZ";
328   return "OTHER";
329 }
330
331 ////////////////////////////////////////////////
332 //
333 // user functions
334
335 function jsrsArrayFromString( s, delim ){
336   // rebuild an array returned from server as string
337   // optional delimiter defaults to ~
338   var d = (delim == null)? '~' : delim;
339   return s.split(d);
340 }
341
342 function jsrsDebugInfo(){
343   // use for debugging by attaching to f1 (works with IE)
344   // with onHelp = "return jsrsDebugInfo();" in the body tag
345   var doc = window.open().document;
346   doc.open;
347   doc.write( 'Pool Size: ' + jsrsContextPoolSize + '<br><font face="arial" size="2"><b>' );
348   for( var i in jsrsContextPool ){
349     var contextObj = jsrsContextPool[i];
350     doc.write( '<hr>' + contextObj.id + ' : ' + (contextObj.busy ? 'busy' : 'available') + '<br>');
351     doc.write( contextObj.container.document.location.pathname + '<br>');
352     doc.write( contextObj.container.document.location.search + '<br>');
353     doc.write( '<table border="1"><tr><td>' + contextObj.container.document.body.innerHTML +
    '</td></tr></table>' );
354   }
355   doc.write('</table>');
356   doc.close();
357   return false;
358 }
```

```
 1
 2 package com;
 3
 4 public class RSExample extends RemoteScriptingServlet {
 5    public static String getSubcategories (String catstr) throws Exception
 6    {
 7      // sure, there is a possibility of an exception happening here, but
 8      // RemoteScriptingServlet will catch it and deal with it appropriately
 9
10      int catid = Integer.parseInt(catstr);
11
12      String subcats[][] = new String[][] {
13        new String[] {"Category 0 - Subcategory 0", "Category 0 - Subcategory 1", "Category 0 - Subcategory 2"},
14        new String[] {"Category 1 - Subcategory 0", "Category 1 - Subcategory 1", "Category 1 - Subcategory 2"},
15        new String[] {"Category 2 - Subcategory 0", "Category 2 - Subcategory 1", "Category 2 - Subcategory 2"},
16      };
17
18      String retval = "";
19      for (int i = 0; i < subcats[catid].length; i++) {
20        // build a string with "index,value" pairs separated by semicolons
21        // for demo brevity, we'll assume that no commas or semicolons are present
22        // in the values of subcats[catid][i]
23        retval += i + "," + subcats[catid][i] + ";";
24      }
25
26      return retval;
27    }
28 }
```


```
1
2  package com;
3
4  // eHatcher Solutions, Inc
5
6  import java.lang.reflect.*;
7  import java.io.*;
8  import java.net.URLEncoder;
9  import java.text.*;
10 import java.util.*;
11 import javax.servlet.*;
12 import javax.servlet.http.*;
13
14 /**
15  * Base class for remote scripting servlets that use Microsoft's remote scripting (MSRS) or JavaScript remote
   scripting (JSRS).
16  *
17  * Usage:
18  * <pre>
19  * public class RSExample extends RemoteScriptingServlet {
20  *    public static String getSubcategories (String catstr) throws Exception
21  *    {
22  *      String retval = "";
23  *        // ... implementation details
24  *      return retval;
25  *    }
26  * }
27  * </pre>
28  *
29  * @version      1.0  February 10, 2001
30  * @author       Erik Hatcher
31  */
32
33 abstract public class RemoteScriptingServlet extends HttpServlet {
34
35    boolean debugOn = false;
36
37    private final static int MSRS = 0;
38    private final static int JSRS = 1;
39
40    private int clientType;
41    public void doPost(HttpServletRequest request, HttpServletResponse response)
42    throws ServletException, IOException {
43       //processRequest(request, response);
44       debugOn = request.getParameter("debug") != null;
45
46       boolean error = false;
47       String returnValue;
48       String callbackName = "";
49
50       // Everything is wrapped in a try/catch block, any exception will cause the ERROR return value
51       // so that the client side can deal with it gracefully
52       try {
53         String method;
54         int pcount = 0;
55
56         callbackName = request.getParameter("C");
57
58
59          // client is JSRS - it passes a "C" parameter
60         //obtain
61          clientType = JSRS;
```

```
62          method = request.getParameter("F");
63
64          // JS call doesn't tell us how many parameters, so count them
65          //get parameters
66          while (request.getParameter("P" + pcount) != null)
67            pcount++;
68
69        // build paramSpec array with all String class items
70        // build params array with the p0, p1, ...., pN request values
71        Class paramSpec[] = new Class[pcount];
72        Class stringClass = Class.forName("java.lang.String");
73        String params[] = new String[pcount];
74        if (pcount > 0) {
75          for (int i=0; i < pcount; i++) {
76            paramSpec[i] = stringClass;
77            params[i] = request.getParameter("P" + i);
78
79            //strip brackets off
80            params[i] = params[i].substring(1, params[i].length() - 1);
81          }
82        }
83
84        // find and invoke the appropriate static method in the concrete class
85        Class c = this.getClass();
86        Method m = c.getMethod(method, paramSpec);
87        returnValue = (String) m.invoke(null, params);
88      } catch (Exception e) {
89        // if the invoked method threw an exception, pull it out of the wrapper exception that "invoke" throws
90        // so that the client gets the real error message
91        if (e instanceof InvocationTargetException) {
92          e = (Exception) ((InvocationTargetException)e).getTargetException();
93        }
94        debug("Oops, exception: " + e);
95        error = true;
96        returnValue = e.toString();
97        e.printStackTrace();
98      }
99      //build up the ourput sting
100     String outputString = "";
101       // Build the appropriate JSRS response
102       outputString = "<html><head></head><body onload=\"p=document.layers?parentLayer:window.parent;";
103       if (error) {
104         outputString += "p.jsrsError('" + callbackName + "','jsrsError: " + encode(returnValue) + "');\">jsrsError: "
    + jsrsErrorEscape(returnValue);
105       }
106       else {
107         outputString += "p.jsrsLoaded('" + callbackName + "');\">jsrsPayload:<br><form
    name=\"jsrs_Form\"><textarea name=\"jsrs_Payload\">" + jsrsEscape(returnValue) + "</textarea></form>";
108         outputString += "</body></html>";
109     }
110     response.setContentType("text/html");
111     PrintWriter out = response.getWriter();
112     out.println(outputString);
113   }
114   /**
115    * Generates the appropriate response to either an MSRS or JSRS method invocation.
116    */
117   public void doGet(HttpServletRequest request, HttpServletResponse response)
118     throws IOException, ServletException
119   {
120     // Requests look like this:
121     // MSRS: <servletname>?_method=method&_mtype=execute&pcount=1&p0=1
```

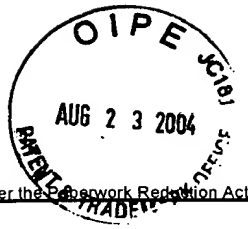
```
122       // JSRS: <servletname>?C=callback&F=method&P0=[1]
123       debugOn = request.getParameter("debug") != null;
124       debug("-------");
125
126      //  response.getWriter().write("dsfsd f sdf sdf ");
127
128       boolean error = false;
129       String returnValue;
130       String callbackName = "";
131
132       // Everything is wrapped in a try/catch block, any exception will cause the ERROR return value
133       // so that the client side can deal with it gracefully
134       try {
135         String method;
136         int pcount = 0;
137
138         callbackName = request.getParameter("C");
139         if (callbackName != null) {
140           // client is JSRS - it passes a "C" parameter
141           clientType = JSRS;
142           method = request.getParameter("F");
143
144           // JSRS doesn't tell us how many parameters, so count them
145           while (request.getParameter("P" + pcount) != null)
146             pcount++;
147         }
148         else {
149           clientType = MSRS;
150           method = request.getParameter("_method");
151           pcount = Integer.parseInt(request.getParameter("pcount"));
152         }
153
154         debug("clientType = " + clientType);
155         debug("Method = " + method);
156         debug("pcount = " + pcount);
157
158
159         // build paramSpec array with all String class items
160         // build params array with the p0, p1, ...., pN request values
161         Class paramSpec[] = new Class[pcount];
162         Class stringClass = Class.forName("java.lang.String");
163         String params[] = new String[pcount];
164         if (pcount > 0) {
165           for (int i=0; i < pcount; i++) {
166             paramSpec[i] = stringClass;
167             params[i] = request.getParameter(( (clientType == MSRS) ? "p" : "P") + i);
168
169             if (clientType == JSRS) {
170               // JSRS sends parameters wrapped with brackets, strip them off
171               params[i] = params[i].substring(1, params[i].length() - 1);
172             }
173
174             debug("p" + i + " = " + params[i]);
175           }
176         }
177
178         // find and invoke the appropriate static method in the concrete class
179         Class c = this.getClass();
180         Method m = c.getMethod(method, paramSpec);
181         returnValue = (String) m.invoke(null, params);
182       } catch (Exception e) {
183         // if the invoked method threw an exception, pull it out of the wrapper exception that "invoke" throws
```

```
184         // so that the client gets the real error message
185         if (e instanceof InvocationTargetException) {
186           e = (Exception) ((InvocationTargetException)e).getTargetException();
187         }
188         debug("Oops, exception: " + e);
189         error = true;
190         returnValue = e.toString();
191         e.printStackTrace();
192       }
193
194       String outputString = "";
195       if (clientType == MSRS) {
196         // Build the appropriate MSRS response
197         // Microsoft's Remote Scripting has three types: SIMPLE, EVAL_OBJECT, and ERROR.
198         // Currently only SIMPLE and ERROR are supported.
199         outputString = "<METHOD VERSION=\"1.0.8044\"><RETURN_VALUE TYPE=" + (error ? "ERROR" :
    "SIMPLE") + ">" + encode(returnValue) + "</RETURN_VALUE></METHOD>";
200       }
201       else {
202         // Build the appropriate JSRS response
203         outputString = "<html><head></head><body onload=\"p=document.layers?parentLayer:window.parent;";
204         if (error) {
205           outputString += "p.jsrsError('" + callbackName + "','jsrsError: " + encode(returnValue) + "');\">jsrsError: "
    + jsrsErrorEscape(returnValue);
206         }
207         else {
208           outputString += "p.jsrsLoaded('" + callbackName + "');\">jsrsPayload:<br><form
    name=\"jsrs_Form\"><textarea name=\"jsrs_Payload\">" + jsrsEscape(returnValue) + "</textarea></form>";
209         }
210         outputString += "</body></html>";
211       }
212
213       response.setContentType("text/html");
214       PrintWriter out = response.getWriter();
215       out.println(outputString);
216     }
217
218     private void debug (String str)
219     {
220       if (debugOn) System.out.println("[RemoteScriptingServlet] " + str);
221     }
222
223     public static String jsrsEscape (String str)
224     {
225       StringBuffer sb = new StringBuffer(str);
226
227       // probably an easier way to do this, but this will do for now
228       for (int i = 0; i < sb.length(); i++) {
229         if (sb.charAt(i) == '/') {
230           sb.replace(i,i+1,"\\/");
231           i += 2;
232         }
233       }
234
235       return new String(sb);
236     }
237
238     public static String jsrsErrorEscape (String str)
239     {
240       StringBuffer sb = new StringBuffer(str);
241
242       // probably an easier way to do this, but this will do for now
```

```
243       for (int i = 0; i < sb.length(); i++) {
244         if (sb.charAt(i) == '\') {
245           sb.replace(i,i+1,"\\");
246           i += 2;
247         }
248         if (sb.charAt(i) == '"') {
249           sb.replace(i,i+1,"\\\"");
250           i += 2;
251         }
252       }
253
254       return new String(sb);
255     }
256
257
258     public static String encode (String str)
259     {
260       // but URLEncoder.encode isn't enough... '+' should really be "%20"
261       StringBuffer sb = new StringBuffer(URLEncoder.encode(str));
262
263       for (int i = 0; i < sb.length(); i++) {
264         if (sb.charAt(i) == '+') {
265           sb.replace(i,i+1,"%20");
266           i += 2;
267         }
268       }
269
270       return new String(sb);
271     }
272  /*
273  MSRS return details:
274
275  String return:
276  <METHOD VERSION="1.0.8044"><RETURN_VALUE
     TYPE=SIMPLE>%3CMessages%3E%3CMsg7%20ID%3D%221%22%3EMsg7%20data%3C/Msg7%3E%3C/
     Messages%3E</RETURN_VALUE></METHOD>
277
278  Error return:
279  <METHOD VERSION="1.0.8044"><RETURN_VALUE
     TYPE=ERROR>xyz%20%3A%20not%20a%20public%20function</RETURN_VALUE></METHOD>
280  */
281
282  /*
283  JSRS return details:
284
285  Successful JSRS return:
286  <html><head></head><body
     onload="p=document.layers?parentLayer:window.parent;p.jsrsLoaded('jsrs1');">jsrsPayload:<br><form
     name="jsrs_Form"><textarea name="jsrs_Payload">string~TEST</textarea></form></body></html>
287
288  Error JSRS return:
289  <html><head></head><body
     onload="p=document.layers?parentLayer:window.parent;p.jsrsError('jsrs1','error');">error</body></html>
290  */
291
292  }
```

41

2126



PTO/SB/21 (02-04)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/916,252
Filing Date	July 30, 2001
First Named Inventor	GAO, YANG
Art Unit	2126
Examiner Name	St. John, Courtney
Attorney Docket Number	

Total Number of Pages in This Submission

ENCLOSURES *(Check all that apply)*

- [X] Fee Transmittal Form
  - [X] Fee Attached
- [X] Amendment/Reply
  - [ ] After Final
  - [X] Affidavits/declaration(s)
- [X] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Response to Missing Parts/ Incomplete Application
  - [ ] Response to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) _____
- [ ] After Allowance communication to Technology Center (TC)
- [ ] Appeal Communication to Board of Appeals and Interferences
- [ ] Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- [ ] Proprietary Information
- [ ] Status Letter
- [ ] Other Enclosure(s) (please identify below):

Remarks

RECEIVED
AUG 30 2004
Technology Center 2100

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	YANG GAO
Signature	
Date	Aug 5, 2004

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Typed or printed name	YANG GAO		
Signature		Date	Aug 5, 2004

This collection of information is required by 37 CFR 1.5. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/17 (10-03)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

FEE TRANSMITTAL for FY 2004

Effective 10/01/2003. Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($)

Complete if Known	
Application Number	09/916,252
Filing Date	30 July 2001
First Named Inventor	GAO, YANG
Examiner Name	St. John Courtenay
Art Unit	2126
Attorney Docket No.	

METHOD OF PAYMENT *(check all that apply)*

☐ Check ☐ Credit card ☐ Money Order ☐ Other ☐ None

☐ Deposit Account:

Deposit Account Number

Deposit Account Name

The Director is authorized to: *(check all that apply)*

☐ Charge fee(s) indicated below ☐ Credit any overpayments

☐ Charge any additional fee(s) or any underpayment of fee(s)

☐ Charge fee(s) indicated below, **except for the filing fee** to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1001	770	2001	385	Utility filing fee	
1002	340	2002	170	Design filing fee	
1003	530	2003	265	Plant filing fee	
1004	770	2004	385	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	

SUBTOTAL (1) ($)

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

	Extra Claims		Fee from below	Fee Paid
Total Claims		-20** =		X =
Independent Claims		- 3** =		X =
Multiple Dependent				=

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	86	2201	43	Independent claims in excess of 3
1203	290	2203	145	Multiple dependent claim, if not paid
1204	86	2204	43	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($)

**or number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for *ex parte* reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	110	2251	55	Extension for reply within first month	55
1252	420	2252	210	Extension for reply within second month	
1253	950	2253	475	Extension for reply within third month	
1254	1,480	2254	740	Extension for reply within fourth month	
1255	2,010	2255	1,005	Extension for reply within fifth month	
1401	330	2401	165	Notice of Appeal	
1402	330	2402	165	Filing a brief in support of an appeal	
1403	290	2403	145	Request for oral hearing	
1451	1,510	1451	1,510	Petition to institute a public use proceeding	
1452	110	2452	55	Petition to revive - unavoidable	
1453	1,330	2453	665	Petition to revive - unintentional	
1501	1,330	2501	665	Utility issue fee (or reissue)	
1502	480	2502	240	Design issue fee	
1503	640	2503	320	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	770	2809	385	Filing a submission after final rejection (37 CFR 1.129(a))	
1810	770	2810	385	For each additional invention to be examined (37 CFR 1.129(b))	
1801	770	2801	385	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid SUBTOTAL (3) ($) 55.00

SUBMITTED BY				(Complete *(if applicable)*)	
Name (Print/Type)	GAO, YANG	Registration No. (Attorney/Agent)		Telephone	(86)-755-26743318
Signature				Date	Aug. 5th. 2004

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.